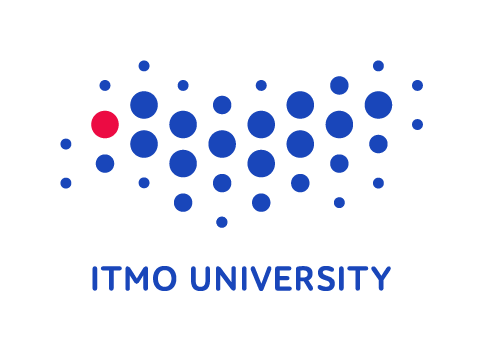
Санкт-Петербургский Национальный Исследовательский

Университет Информационных Технологий, Механики и Оптики

Факультет Программной Инженерии и Компьютерной Техники



Варианты №: 1019

Лабораторная работа № 5

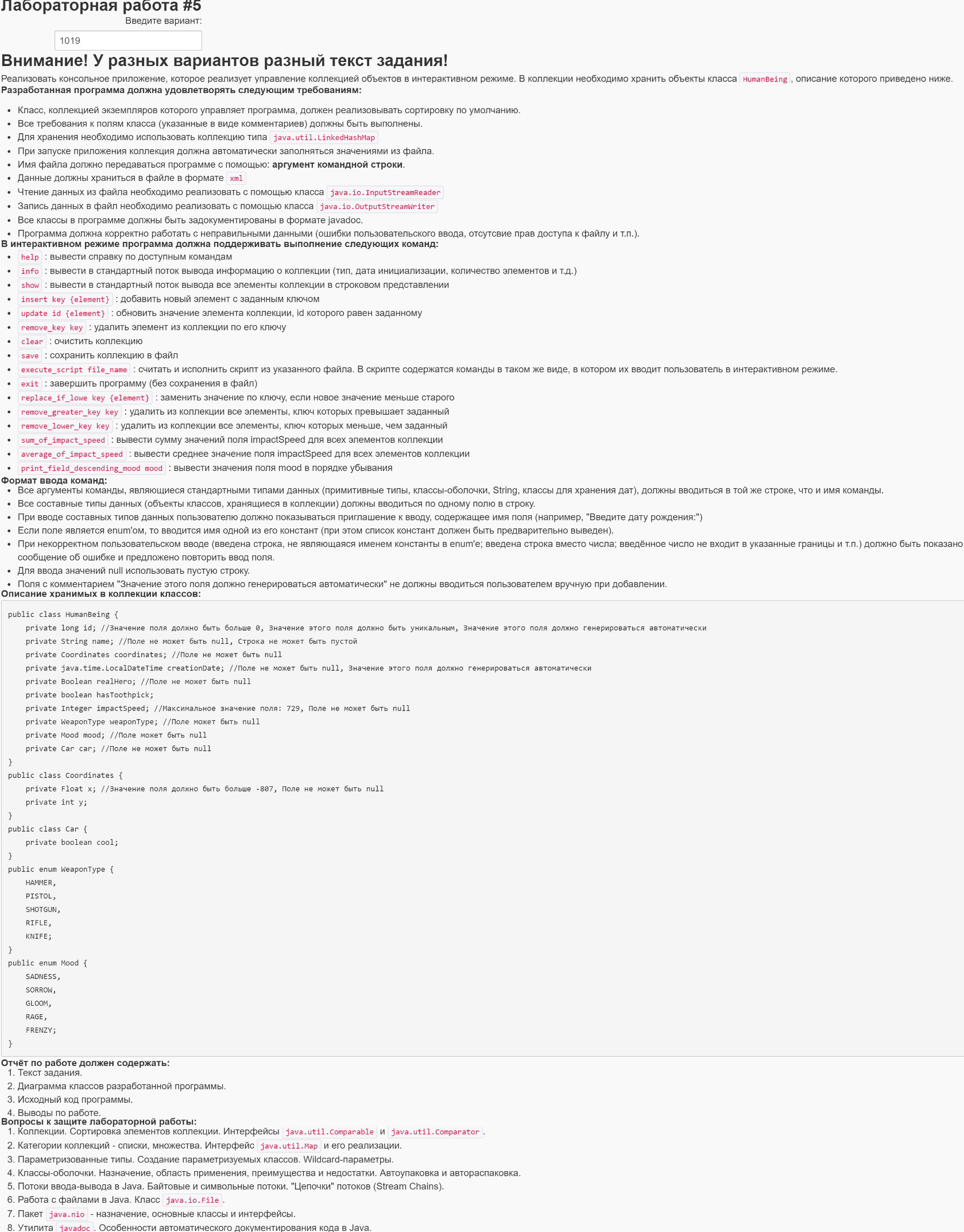
по дисциплине

*«Программирование»*

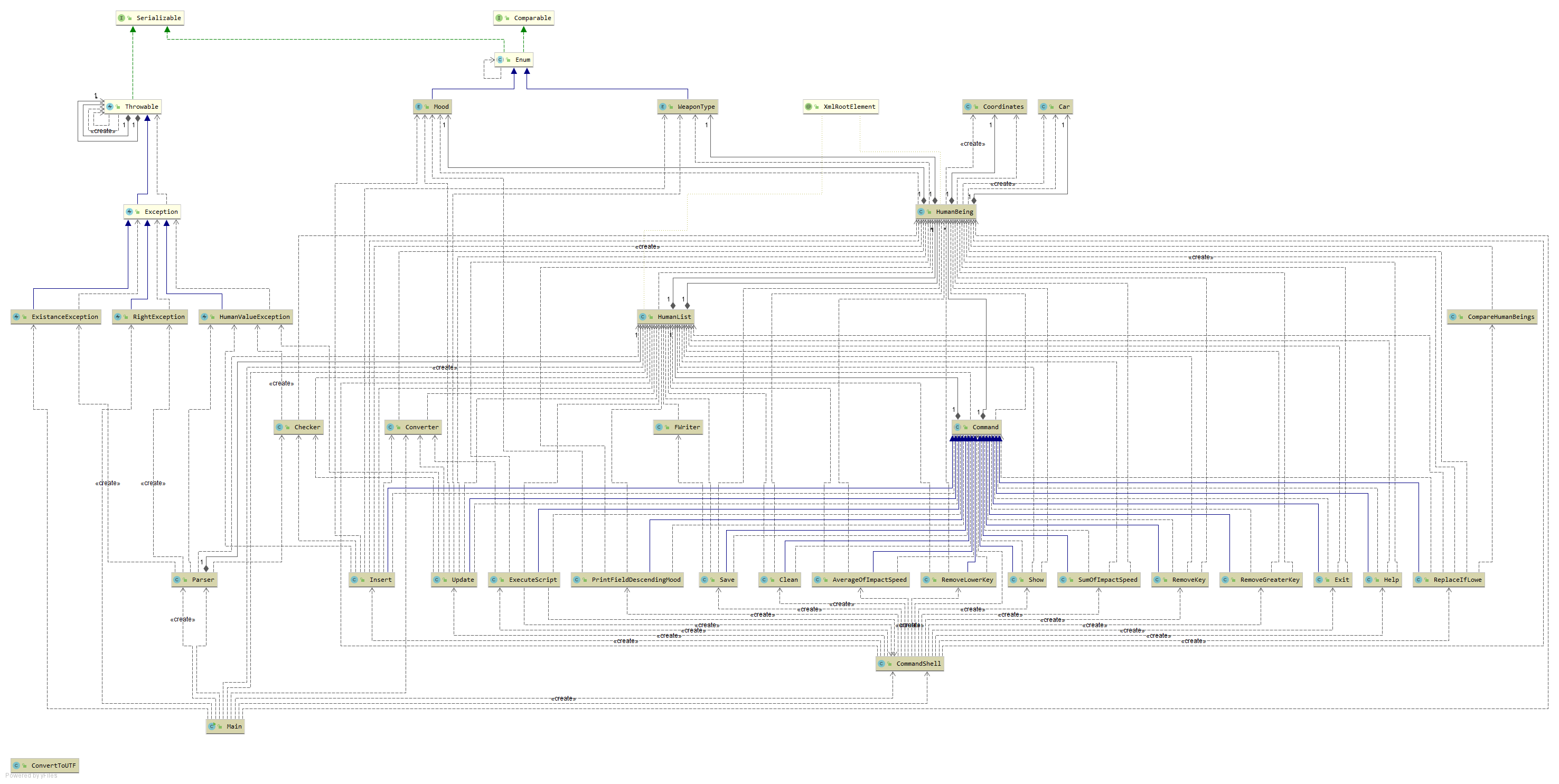
Выполнил: Провоторов Александр Владимирович  
Студент группы: P3110

Преподаватель: Николаев Владимир Вячеславович

# ****Текст Задания****



# ****Диаграмма классов реализованной объектной модели.****

****

# ****Исходный код программы.****

## ****Class Main****

**import** MyOwnClasses.CommandShell;  
**import** MyOwnClasses.HumanBeing;  
**import** MyOwnClasses.HumanList;  
**import** Tools.Parser;  
  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
**import** java.util.\*;  
  
**import static** Tools.Converter.*convertToList*;  
**import static** Tools.Converter.*сonvertToLinkedHashMap*;  
  
**public class** Main {  
 **public static void** main(String []args) **throws** IOException {  
 InputStreamReader reader = **new** InputStreamReader(System.***in***);  
 BufferedReader bufferedReader = **new** BufferedReader(reader);  
  
 *//In this block first we parse xml to HumanList and then to LinkedHashMap* Parser Parser = **new** Parser();  
 Parser.parse();  
 LinkedHashMap<Integer, HumanBeing> human = *сonvertToLinkedHashMap*((Parser.getHumanList()));  
 HumanList humanList = **new** HumanList();  
 humanList.setHumanBeings(*convertToList*(human));  
  
 **boolean** b = **true**;  
 **while** (b || reader.ready()) {  
 String comand = bufferedReader.readLine(); *//read command* human = CommandShell.*Analyze*(humanList, human, comand); *//processing command and equalizing the result to LinkedHashMap* humanList.setHumanBeings(*convertToList*(human)); *//LinkedHashMap -> HumanList* }  
 }  
}

## Class Parser

**package** Tools;  
  
**import** Exceptions.HumanValueException;  
**import** MyOwnClasses.HumanList;  
  
**import** javax.xml.bind.JAXBContext;  
**import** javax.xml.bind.JAXBException;  
**import** javax.xml.bind.Unmarshaller;  
**import** java.io.File;  
  
**import static** Tools.Checker.*checkHumanList*;  
  
  
**public class** Parser{  
  
 **private** HumanList **humanList**;  
 **public void** parse() {  
 **try** {  
 File file = **new** File(**"Exercise.xml"**);  
 JAXBContext jaxbContext = JAXBContext.*newInstance*(HumanList.**class**);  
  
 Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();  
  
 HumanList humanList = (HumanList) jaxbUnmarshaller.unmarshal(file);  
  
 **this**.**humanList** = humanList;  
  
 } **catch** (JAXBException e) {  
 }  
  
 **for** (**int** i = 0; i < **humanList**.getHumanBeings().size(); i++) **humanList**.getHumanBeing(i).setId(i+1);  
  
 **try** {  
 *checkHumanList*(**humanList**);  
 } **catch** (HumanValueException e){  
 e.printStackTrace();  
 }  
 }  
 **public** HumanList getHumanList(){  
 **return humanList**;  
 }  
}

## Class ComandShell

**package** Command;  
  
**import** Exceptions.HumanValueException;  
**import** MyOwnClasses.HumanBeing;  
**import** MyOwnClasses.HumanList;  
  
**import** java.util.\*;  
  
**public class** CommandShell {  
 **public** CommandShell() {}  
 **public** LinkedHashMap<Integer, HumanBeing> Analyze(HumanList humanList, LinkedHashMap<Integer, HumanBeing> human, String command, String path, **boolean** b) {  
 HashMap<String, Command> commandList = **new** LinkedHashMap<>();  
 StringTokenizer tokenizer = **new** StringTokenizer(command);  
 commandList.put(**"save"**, **new** Save(human, command, humanList));  
 commandList.put(**"info"**, **new** Info(human, command, humanList));  
 commandList.put(**"exit"**, **new** Exit(human, command, humanList));  
 commandList.put(**"help"**, **new** Help(human, command, humanList));  
 commandList.put(**"show"**, **new** Show(human, command, humanList));  
 commandList.put(**"clean"**, **new** Clean(human, command, humanList));  
 commandList.put(**"update"**, **new** Update(human, command, humanList));  
 commandList.put(**"insert"**, **new** Insert(human, command, humanList));  
 commandList.put(**"remove\_key"**, **new** RemoveKey(human, command, humanList));  
 commandList.put(**"execute\_script"**, **new** ExecuteScript(human, command, humanList));  
 commandList.put(**"replace\_if\_lowe"**, **new** ReplaceIfLowe(human, command, humanList));  
 commandList.put(**"remove\_lower\_key"**, **new** RemoveLowerKey(human, command, humanList));  
 commandList.put(**"remove\_greater\_key"**, **new** RemoveGreaterKey(human, command, humanList));  
 commandList.put(**"sum\_of\_impact\_speed"**, **new** SumOfImpactSpeed(human, command, humanList));  
 commandList.put(**"average\_of\_impact\_speed"**, **new** AverageOfImpactSpeed(human, command, humanList));  
 commandList.put(**"print\_field\_descending\_mood"**, **new** PrintFieldDescendingMood(human, command, humanList));  
 **try** {  
 String token = tokenizer.nextToken();  
 human = commandList.get(token.toLowerCase()).execute(human, command, humanList, path, b);  
 } **catch** (StringIndexOutOfBoundsException e) { System.***out***.println(**"Команда была введена неверно"**); }  
 **catch** (ArithmeticException e) { System.***out***.println(**"Лист HumanList пуст"**); }  
 **catch** (NullPointerException e) { System.***out***.println(**"Команда была введена неверно"**); }  
 **catch** (HumanValueException e) { System.***out***.println(**"Введите правильный тип данных"**); }  
 **return** human;  
 }  
}

# ****Результат работы программы.****

|  |
| --- |
| **show** |
| **Коллекция: {0=HumanBeing{id=1, name='Я', coordinates=MyOwnClasses.Coordinates@69d0a921, creationDate=2020-02-21T01:44:25.172, realHero=true, hasToothpick=true, impactSpeed=20, weaponType=SHOTGUN, mood=RAGE, car=MyOwnClasses.Car@446cdf90}, 1=HumanBeing{id=2, name='Футбольный', coordinates=MyOwnClasses.Coordinates@799f7e29, creationDate=2020-02-21T01:44:25.174, realHero=true, hasToothpick=true, impactSpeed=20, weaponType=SHOTGUN, mood=FRENZY, car=MyOwnClasses.Car@4b85612c}, 2=HumanBeing{id=3, name='Мячик', coordinates=MyOwnClasses.Coordinates@277050dc, creationDate=2020-02-21T01:44:25.174, realHero=true, hasToothpick=true, impactSpeed=20, weaponType=SHOTGUN, mood=SADNESS, car=MyOwnClasses.Car@5c29bfd}}** |
| **update 1** |
| **Введите имя: I** |
| **Введите координату X: 11** |
| **Введите координату Y: 11** |
| **Является ли персонаж настоящим героем? (true/false)** |
| **false** |
| **Имеет ли персонаж зубную щетку? (true/false)** |
| **false** |
| **Введите скорость удара:** |
| **2** |
| **Введите оружие персонажа из предложенных (Hammer Pistol Shotgun Rifle Knife):** |
| **Hammer** |
| **Введите настроение персонажа из предложенных (Sadness Sorrow Gloom Rage Frenzy):** |
| **Sadness** |
| **По настоящему ли крута машина персонажа (true/false):** |
| **false** |
| **show** |
| **Коллекция: {0=HumanBeing{id=4, name='I', coordinates=MyOwnClasses.Coordinates@69d0a921, creationDate=2020-02-21T01:44:25.172, realHero=false, hasToothpick=false, impactSpeed=2, weaponType=HAMMER, mood=SADNESS, car=MyOwnClasses.Car@446cdf90}, 1=HumanBeing{id=2, name='Футбольный', coordinates=MyOwnClasses.Coordinates@799f7e29, creationDate=2020-02-21T01:44:25.174, realHero=true, hasToothpick=true, impactSpeed=20, weaponType=SHOTGUN, mood=FRENZY, car=MyOwnClasses.Car@4b85612c}, 2=HumanBeing{id=3, name='Мячик', coordinates=MyOwnClasses.Coordinates@277050dc, creationDate=2020-02-21T01:44:25.174, realHero=true, hasToothpick=true, impactSpeed=20, weaponType=SHOTGUN, mood=SADNESS, car=MyOwnClasses.Car@5c29bfd}}** |
| **save** |
| **Коллекция сохранена** |
| **exit** |
|  |
| **Process finished with exit code 1** |

# ****Вывод****

**В ходе выполнения лабораторной работы я познакомился с некоторыми видами коллекций, категориями коллекций. Также затронул потоки ввода-вывода, поработал с файлами в Java и попробовал задокументировать файл в Javadoc.**